Certificate Task: MERN Stack Social Networking Application

Objective

Build a functional Social Networking Application using the MERN (MongoDB, Express.js, React.js, Node.js) stack. The application should include CRUD operations for posts, demonstrate your ability to integrate frontend and backend technologies, and incorporate user authentication.

Task Description

Develop a Mini Social Networking Platform where users can create, view, and interact with posts, manage their profiles, and connect with other users.

Project Requirements

Frontend (React.js)

1. **User Interface (UI):**
    - Create a responsive and user-friendly UI with components for:
        - Listing all posts.
        - Viewing a single post with its details.
        - Creating a new post (authenticated users only).
        - Editing or deleting posts (post owner only).
        - Viewing and updating user profiles.
        - Searching for other users.
2. **Features:**
    - Display posts in a feed format, including:
        - Post Content
        - Author Name
        - Date of Creation
        - Number of Likes and Comments
    - A profile page for each user that displays:
        - Profile Picture
        - Username
        - Bio
        - User’s Posts
    - A form to create/edit posts with validation.
    - A search bar to find posts or users.
3. **State Management:**

    - Use React Context API or a state management library like Redux.
4. **Styling:**
    - Use Bootstrap, Material-UI, or custom CSS for a professional-looking interface.

Backend (Node.js, Express.js)

1. **API Development:**
    - Develop a RESTful API with the following endpoints:
        - `POST /posts`: Add a new post (authenticated users only).
        - `GET /posts`: Retrieve all posts.
        - `GET /posts/:id`: Retrieve a specific post by ID.
        - `PUT /posts/:id`: Update post details (post owner only).
        - `DELETE /posts/:id`: Delete a post (post owner only).
        - `GET /users/:id`: Retrieve a specific user's profile.
        - `PUT /users/:id`: Update user profile (profile owner only).
2. **Middleware:**
    - Validate incoming data using Joi or express-validator.
    - Implement role-based and owner-based access control (e.g., only the post owner can edit or delete their posts).
3. **Error Handling:**
    - Handle common errors like invalid requests, non-existent resources, and unauthorized actions.

Database (MongoDB)

1. **Database Design:**
    - Create MongoDB collections for:
        - `posts` with fields:
            - `_id` (auto-generated by MongoDB)
            - `content` (String, required)
            - `author` (User reference, required)
            - `likes` (Array of User references)
            - `comments` (Array of objects with comment text, author, and timestamp)
            - `createdAt` (Date, default: current date)
        - `users` with fields:
            - `_id` (auto-generated by MongoDB)
            - `username` (String, required)
            - `password` (Hashed string, required)

        - `profilePicture` (String, optional)
        - `bio` (String, optional)
2. **Connection:**
    - Use Mongoose for schema definition and database interaction.
    - Configure the application to connect to a MongoDB instance (local or cloud-based like MongoDB Atlas).

Deployment

1. Deploy the application to:
    - **Frontend:** Use Netlify or Vercel.
    - **Backend:** Use Render, Heroku, or AWS.
    - **Database:** Use a cloud MongoDB database (e.g., MongoDB Atlas).
2. Provide the deployed application link along with a GitHub repository containing the source code.

Bonus Features (Optional)

- **User Authentication:**
    - Implement user registration and login with hashed passwords and JWT-based authentication.
- **Follow/Unfollow System:**
    - Enable users to follow and unfollow each other, displaying a follower/following count on profiles.
- **Post Likes and Comments:**
    - Add functionality to like posts and leave comments.
- **Image Upload:**
    - Allow users to upload profile pictures and images in posts using Multer or a cloud service like Cloudinary.
- **Notification System:**
    - Notify users about likes, comments, or new followers.